Über die Bedeutung eines statischen Kernes für die Sicherheit Mobiler Software-Agenten

Volker Roth

Fraunhofer Institut für Graphische Datenverarbeitung
Rundeturmstraße 6, 64283 Darmstadt, Germany
vroth@igd.fhg.de

Zusammenfassung

In diesem Artikel untersuche und beschreibe ich anhand dreier Bereiche die Bedeutung eines global eindeutigen statischen Kernes für die Sicherheit eines mobilen Agenten gegen Angriffe: der Identifikation mobiler Agenten, der Integrität und Vertraulichkeit von Daten und Berechtigungen in einem mobilen Agenten, sowie der Verfolgung mobiler Agenten unter bestmöglicher Wahrung der Privatsphäre von deren Besitzern.

Keywords: Mobile Agenten, Sicherheit.

1 Einführung

Mobile Agenten Systeme [23] eliminieren die letzte Schranke wahrhaft verteilter Systeme – die Bindung von Programm und Ausführungszustand an einen frei wählbaren aber fixierten Rechner. Sie bilden die Grundlage zukünftiger selbstorganisierender Systeme, in denen autonome Softwarebausteine dynamisch, adaptiv und zielgerichtet auf vorhandene Ressourcen aufgeteilt werden können. Mobile Agenten Systeme an sich sind kein neues Konzept – bereits Anfang der 80'er wurde mit Wurmprogrammen experimentiert [18], die prinzipiell die gleichen Merkmale wie Mobile Agenten aufweisen – nach wie vor bildet jedoch die Frage nach wirkungsvollen Sicherheitsmechanismen die größte Hürde für den Einsatz von Mobile Agenten Systemen. Dies gilt insbesondere bei Anwendungen in offenen Netzen ohne feste Vertrauensverhältnisse, wie es im Internet der Fall ist.

Mittlerweile sind eine Reihe von Verfahren zur Sicherung verschiedener Aspekte von Mobile Agenten Systeme veröffentlicht worden. Eine Sammlung der wichtigsten Arbeiten findet sich in [21, 22, 13]. Nachteilig ist jedoch zu vermerken, daß nur wenige Ansätze bisher gleichermaßen eine hohes Maß an Sicherheit bieten und auch in Bezug auf ihre Komplexität für einen praktischen Einsatz geeignet sind. So beschreibt beispielsweise Vigna [20] ein Protokoll zur Prüfung der korrekten Ausführung von Agenten, der dazu erforderliche Aufwand, die notwendigen Einschränkungen und die Anforderungen an die Mobile Agenten Infrastruktur machen den beschriebenen Ansatz im allgemeinen jedoch unattraktiv. Auf der anderen Seite sind Ansätze hervorzuheben, wie sie von Karjoth et al. [5] auf Basis der Vorarbeiten von Yee [25] beschrieben werden. Deren Ansatz schützt die Integrität der Daten mobiler Agenten mit einem Aufwand, der durchaus praktikabel ist.

Im folgenden fasse ich eine Reihe weiterer Ansätze zusammen, die zur Anwendung gebracht werden können, um die praktische Sicherheit mobiler Agenten gegenüber dem Stand der Technik zu verbessern. Allen diesen Ansätzen ist ein eindeutiger statischer Kern im mobilen Agenten gemein, der als Anker für die erforderlichen kryptographischen Protokolle dient.

Die Bedeutung eines solchen statischen Kerns läß sich insbesondere im Vergleich mit einem

reinen *code signing* Konzept verdeutlichen, wie es bei *Applets* zur Anwendung kommt. Dies ist Gegenstand von Abschnitt 2. In Abschnitt 3 beschreibe ich einen Ansatz zur Verschlüsselung von Daten innerhalb eines mobilen Agenten und motiviere, wie und warum die resultierenden Kryptogramme an den Agenten gebunden werden müssen. Auf die Berechnung von impliziten Namen auf Agenten und deren Anwendung im Bereich der Verfolgung von mobilen Agenten mittels Namensdiensten gehe ich in Abschnitt 4 ein, und schließe mit Abschnitt 5, in dem ich eine Reihe von Schlußfolgerungen ziehe.

2 Signierte Mobile Agenten

Ein mobiler Agent läßt sich prinzipiell in zwei Arten von Daten unterteilen: veränderliche Daten und statische Daten. Letztere bleiben wärend der Lebenszeit des Agenten konstant, wohingegen die veränderlichen Daten den jeweils letzten Zustand eines Agenten repräsentieren, und damit auch die Informationen, die der Agent auf seinem Weg erwirbt. Zu den statischen Daten gehört im allgemeinen das ausführbare Agentenprogramm[1].

Die Ausführung eines Agenten erfolgt üblicherweise durch Interpretation des Agentenprogrammes durch das Wirtssystem, die Eingabe des Agentenprogrammes besteht aus den mitgeführten Daten eines Agenten und externen Daten, die durch das Wirtssystem bereitgestellt oder vermittelt werden.

Hier offenbart sich bereits ein fundamentales Problem der Sicherheit von mobilen Agenten. Die Wahl der Interpretation wird durch das Wirtssystem kontrolliert. Zwar legt beispielsweise die Versendung eines in *Java* programmierten Agenten die Wahl einer Interpretation nahe, die auf der offiziellen Spezifikation der *Java Virtual Machine* beruht. Dies ist jedoch aus Sicht des Agentennutzers eine einseitige Vereinbarung, ein Wirtssystem ist prinzipiell durch nichts als gutem Willen an diese Spezifikation gebunden. Auch über die Qualität und den Ursprung der Eingaben, auf denen das Agentenprogramm interpretiert wird, können a-priori keine Annahmen getroffen werden.

Dies führt dazu, daß die *Identitätsannahme* – die traditionell im Bereich der Zugangskontrolle Anwendung findet – ab dem zweiten Sprung eines mobilen Agenten nicht mehr greift. Diese Annahme bezieht sich darauf, daß Daten und Operationen Entitäten zugeordnet werden können, deren Identität angegeben werden kann, und daß eine Entität, der eine Operation zugeordnet werden kann, auch wirklich die Intention hatte, diese Operation durchzuführen (oder durchführen zu lassen) [1].

Die automatische Evaluierung des Agentenzustandes anhand eines zertifizierten Teilprogrammes, wie dies vorgeschlagen wurde [3] um letztendlich die Identitätsannahme zu stützen, läßt sich nicht verallgemeinern, was am Beispiel des *Halte-Problems* leicht nachvollzogen werden kann [2].

Gemeinhin behilft man sich damit, daß das Agentenprogramm vom Besitzer des Agenten digital signiert wird. Allerdings kann eine digitale Signatur leicht zusammen mit dem zugehörigen Programm gestohlen und mißbraucht werden, wenn Privelegien an Agenten lediglich auf Basis signierter Programme vergeben werden. Dies ist insbesondere dann kritisch, wenn multiple Signaturen und Ensembles aus Softwarekomponenten unterstützt werden, wie dies in Java der Fall ist. Die wichtige Erkenntnis ist, daß der Besitzer eines Agenten die Ausführung eines Agentenprogrammes *nur in einem festgelegten Kontext autorisiert* – der Ausführung einer ausgezeichneten Instanz seines Agenten.

Mit anderen Worten, der Besitzer eines mobilen Agenten sollte zwar sehr wohl dessen Programm

[1] Sofern man keine selbstmodifizierenden Agenten zuläßt.

signieren, aber zusammen mit statischen Angaben, die eine eindeutige Instanz seines Agenten identifizieren, beispielsweise: der eindeutige Name des Agenten, seine Gültigkeitsdauer, die maximal für den Agenten beantragten Rechte, eine Adresse, an die Rückmeldungen des Agenten geschickt werden dürfen und dergleichen mehr. Dies stellt sicher, daß das Agentenprogramm mit den Rechten, die ihm auf Basis der digitalen Unterschrift des Agentenbesitzers zugewiesen werden, nicht in einem anderen Kontext mißbraucht werden oder mehr Rechte als vorgesehen erwerben können.

Dies legt nahe, die Daten eines Agenten auf eine Weise zu strukturieren, die kryptographische Operationen auf dem Agenten getrennt nach statischen und veränderlichen Daten unterstützt. Wärend der Ausführung eines Agenten sollte dieser lesenden Zugriff auf alle seine Daten und schreibenden Zugriff auf seine veränderlichen Daten erhalten. Eine strukturierte Darstellung von Agentendaten ermöglicht darüberhinaus die transparente Bereitstellung von kryptographischen und anderen Diensten für mobile Agenten, wie dies noch näher in Abschnitt 3 beschrieben wird. Ansätze einer solchen Struktur habe ich bereits in [12] beschrieben, eine aktuellere Darstellung findet sich in [11].

Für den vorliegenden Artikel ist jedoch lediglich von Bedeutung, daß ein mobiler Agent einen statischen Kern beinhaltet, durch den sein Ausführungskontext, seine Identität, sein Programm, seine beantragten Privelegien und dergleichen mehr bestimmt sind. Daten, die ein Agent erwirbt oder mit sich führt, müssen nachvollziehbar an diesen statischen Kern gebunden werden, um *cut & paste* Angriffe zu verhindern.

3 Vertrauliche Daten in Mobilen Agenten

Je nach Anwendung ist es wünschenswert, daß ein mobiler Agent zumindestens Teile seiner Daten vor seinem aktuellen Wirtssystem geheim hält. Wenn eine lokal determinierte Folge von Zustandsänderungen des Agenten existiert, die dazu führt, daß der Agent auf den Klartextdaten operiert, dann kann das Wirtssystem in deren Besitz gelangen, indem es diese Folge simuliert. Dies folgt einfach aus dem Umstand, daß das Wirtssystem ein Universalprogramm für die Agentenprogramme darstellt. Für die Auswertungen von Polynomen existieren allerdings bereits Ansätze auf Basis homomorpher Kryptosysteme, welche die Geheimhaltung der berechneten Funktion vor dem Wirtssystem erlauben [16].

Rückt man etwas von der eingangs genannten Forderung ab und verlangt statt dessen, daß gewisse Daten eines Agenten nur auf bestimmten Wirtssystemen eingesehen werden können, dann kann das im folgenden beschriebene Vorgehen gewählt werden.

Man verschlüsselt die geheim zu haltenden Daten mit einem zufällig gewählten Schlüssel k unter Verwendung einer symmetrischen Chiffre. Der Schlüssel k wird nun mit den öffentlichen Schlüsseln derjenigen (autorisierten) Wirtssysteme verschlüsselt, auf denen die Daten zugreifbar sein sollen. Diese *hybride Verschlüsselung* (die Kombination von asymmetrischen und symmetrischen Chiffren) erweist sich in der Praxis als effizienter als eine rein asymmetrische Verschlüsselung und reduziert darüberhinaus die Angriffsmöglichkeiten auf die asymmetrische Chiffre. Eine rein asymmetrische Verschlüsselung der Daten des Agenten verbietet sich abgesehen davon bereits daher, daß in diesem Fall die zu schützenden Daten mehrfach im Agenten gespeichert werden müssten (pro autorisiertem Empfänger einmal, verschlüsselt mit dessem öffentlichen Schlüssel).

Die resultierenden Kryptogramme werden dem Agenten mitgegeben. Als Datenstruktur bietet sich PKCS#7 an [14]. Auf einem autorisierten Wirtssystem sind die Daten daher einsehbar und auch veränderbar, wenn sie nicht dem statischen Teil des Agenten angehören (und somit

der digitalen Signatur des Agenten unterliegen), denn ein autorisiertes Wirtssystem kann die geänderten Daten unter Verwendung von k neu verschlüsseln.

Man sollte meinen, daß damit das Problem gelöst sei. Es existiert jedoch eine einfache Möglichkeit für nicht autorisierte Wirtssysteme in den Besitz des Klartextes der verschlüsselten Daten zu gelangen. Dazu kopiert das Wirtssystem die Kryptogramme des Agenten in einen eigenen Agenten und sendet diesen zu einem autorisierten Wirtssystem *(cut & paste)*. Dort werden die Daten entschlüsselt, als ob sie zu dem betrügerischen Agenten gehörten, der anschließend den Klartext kopiert und zum nicht autorisierten Wirtssystem bringt. Dieser Angriff kann gelingen, *weil die Kryptogramme nicht an einen bestimmten Agenten gebunden sind.*

Das Problem kann auf folgende Weise gelöst werden: der Agentenbesitzer berechnet einen MAC [7, Kap. 9] *(Message Authentication Code)* auf k und dem Zertifikat des Schlüssels, mit dem er seinen Agenten signiert. Diesen MAC kopiert er in den statischen Teil seines Agenten bevor er diesen signiert. Bevor ein Wirtssystem nun das Kryptogramm eines Agenten entschlüsselt, überprüft es die Signatur auf dem statischen Teil des Agenten und unter Verwendung von k und dem Schlüsselzertifikat den in im enthaltenen MAC.

Ein Angreifer kann den MAC nicht in einen eigenen Agenten einbauen, denn dieser ist nicht in Verbindung mit seinem eigenen Zertifikat gültig. Um einen gültigen MAC mit seinem Zertifikat zu berechnen ist wiederum die Kenntnis von k erforderlich. Das Agentenprogramm ist ebenfalls durch den Besitzer des Agenten mit dem statischen Teil zusammen signiert. Außer einem Angriff auf die verwendeten kryptographischen Algorithmen bleibt dem Angreifer nur die Modifikation des veränderlichen Teiles des Agenten. Agenten müssen also sorgfältig programmiert sein, damit ein Export von Klartextdaten von einem autorisierten Wirtssystem durch eine solche Modifikation nicht möglich ist.

Mit anderen Worten: um die verschlüsselten Daten, die ein Agent mit sich führt, von einem Wirtssystem entschlüsseln zu lassen, muß der Besitzer des Agenten durch einen gültigen MAC beweisen, daß er den erforderlichen Schlüssel k kennt. Ein vergleichbares Protokoll läßt sich prinzipiell auch auf Basis digitaler Signaturen aufbauen; die Nutzung eines MAC ist jedoch i.a. effizienter, sowohl was den Speicherbedarf des MAC betrifft, als auch die Anzahl der erforderlichen Rechenoperationen zu dessen Berechnung und Prüfung. Eine detaillierte Beschreibung dieses Protokolles findet sich in [11].

4 Die Verfolgung von Mobilen Agenten

Ein Forschungsbereich, der gegenwärtig verstärktes Interesse auf sich zieht, ist die Bereitstellung *transparenter Kommunikation* zwischen mobilen Agenten [24, 8, 17, 9]. “Transparent” bedeutet in diesem Zusammenhang, daß kommunizierende Agenten nicht der Position ihres Gegenübers gewahr sein müssen; das *routing* und die Zustellung von Nachrichten erfolgt automatisch durch die Mobile Agenten Infrastruktur. Hierbei sind eine Reihe von Problemen zu lösen, wie beispielsweise *guaranteed delivery*, denn mobile Agenten können vor Nachrichten, die an sie gesendet werden, “davonlaufen”. Das *routing* von Nachrichten an mobile Agenten erfordert eine Form von *Namensdienst*, der die eindeutige Kennung eines Agenten auf dessen letzte bekannte Position abbildet.

Einen solchen Namensdienst für ein weltumspannendes Netz wie das Internet zu entwickeln stellt hohe Anforderungen an dessen Skalierbarkeit, aber auch an die Sicherheit gegen Angriffe. Ein zentralisiertes System ist zu diesem Zweck sicherlich nicht geeignet, bei einem verteilten Namensdienst muß die Wahl des *name server*, der für einen Agenten verantwortlich ist, aus dem Namen des Agenten hervorgehen. Diese Wahl muß zudem vom Besitzer des Agenten nachprüfbar

bestätigt werden (z.B. durch eine digitale Signatur), denn ansonsten könnte ein Angreifer möglicherweise Namen von Agenten fälschen oder auch erraten, eigene Agenten unter diesen Namen publizieren, und dadurch ggf. auch reguläre Agenten blockieren *(Denial of Service)*, oder an sie gerichtete Nachrichten abfangen. Als weiterer positiver Nebeneffekt wird die Wahl des Namens durch die Signatur eindeutig.

Dieser Ansatz birgt jedoch auch eine Reihe von Nachteilen. Namensdienste werden omnipotent und "sehen" in aller Regel die Routen der von ihnen verwalteten Agenten. Auch ist genau ersichtlich, welche Agenten zu welchem Benutzer gehören. Es ergeben sich weitreichende Konsequenzen für die Aufweichung der Privatsphäre der Agentenbesitzer, denn ein Agent handelt um so besser für seinen Besitzer, je mehr er dessen Vorlieben und Ziele repräsentiert. Der Erstellung von Benutzerprofilen durch die Betreiber von Namensdiensten ist damit Tür und Tor geöffnet.

Eine Lösung bietet wiederum der statische Kern eines mobilen Agenten, oder etwas genauer, die Signatur des Agenten durch dessen Besitzer. Diese ist nach Annahme bereits eindeutig und nicht fälschbar. Da die Struktur eines Agenten jedoch u.U. einen regelmäßigen Aufbau besitzt, sollte dieser vor der Berechnung der Signatur ggf. ein ausreichendes Maß an zufälligen Daten hinzugefügt werden, z.B. das Datum und Uhrzeit der Erzeugung. Sofern PKCS#7 als Syntax für die Darstellung des Signatur gewählt wird, kann hierfüer beispielsweise auch *Signing Time* als *authenticated attribute* herangezogen werden [14, 15].

Der Name eines Agenten wird nun *implizit* durch Anwendung einer Einweg-Hashfunktion aus der kodierten Signatur gewonnen. Die verwendete Hashfunktion muß *preimage resistant* und *2nd-preimage resistant* [7, S. 323] und die Wahrscheinlichkeit einer zufälligen Kollision muß vernachlässigbar gering sein. Für die Praxis eignet sich z.B. der SHA1 Algorithmus [4]. Der Name eines Agenten wird jeweils durch sein Wirtssystem berechnet, sobald ein migrierender Agent empfangen wird. Die Verwendung impliziter Namen hat eine Reihe von Vorteilen:

- Die Namen können nicht a-priori geraten oder gefälscht werden, außer es wird der private Signaturschlüssel oder das Signaturverfahren oder die Einweg-Hashfunktion gebrochen.
- Der Namensdienst kann die impliziten Namen nicht mit den Besitzern von Agenten in Verbindung bringen (wie dies der Fall wäre, würde die Signatur selbst als Name verwendet werden), es sei denn er kollaboriert mit einem der vergangenen Wirtssysteme eines jeden Agenten, dessen Zuordnung er herausfinden möchte.
- Die Zugehörigkeit von *unterschiedlichen* Agenten zu ihren Besitzern kann nicht aus den Namen korreliert werden.
- Agenten können einem Wirtssystem gegenüber keine falschen Namen angeben, denn das Wirtssystem berechnet den Namen implizit aus der Signatur. Agenten mit unterschiedlichem statischen Teil oder unterschiedlichen Besitzern haben automatisch auch unterschiedliche Namen.
- Die zur Berechnung impliziter Namen verwendeten kryptographischen Operationen sind sehr einfach und effizient. Implizite Namen sind i.a. kürzer als die Signatur aus der sie berechnet werden.

Die Wahl von impliziten Namen für mobile Agenten läßt sich mit der Berechnung von *put-ports* aus *get-ports* für Server-Prozesse im Betriebssystem *Amoeba* vergleichen [19, S. 607]. Eine weitaus detailliertere Diskussion von skalierbaren und sicheren Namensdiensten für mobile Agenten, wie auch weitere Protokolle, sind in [10] zu finden.

Implizite Namen bieten sich auch an, um Daten nachträglich an Agenten zu binden, beispielsweise erweiterte Rechte, die auf nachfolgenden Wirtssystemen von einem Agenten wahrgenommen

werden können. Der Aussteller der Rechte signiert diese dazu zusammen mit dem impliziten Namen des Agenten, dem die Rechte zugesprochen werden sollen. Das verifizierende Wirtssystem prüft später dann die Zuordnung der Rechte zu dem Agenten, der diese in Anspruch nehmen möchte, die Signatur des Ausstellers und ob der Aussteller zur Vergabe der Rechte an diesen Agenten befugt ist (anhand der lokalen Sicherheitspolitik).

5 Schlußfolgerungen

In diesem Artikel habe ich eine Reihe von Sicherheitsmechanismen für mobile Agenten kurz beschrieben, deren Gemeinsamkeit darin besteht, daß sie sich die Existenz eines eindeutigen statischen Kernes pro mobilem Agenten zunutze machen. Dieser Kern dient gleichsam als "Anker" an dem die für seine Ausführung relevanten Daten festgemacht werden. Der Einfluß dieses statischen Kernes ist weitreichend und erstreckt sich über die Erkennung und Abwehr von *cut & paste* Angriffen auf verschlüsselte Daten und Berechtigungen eines Agenten bis hin zum Schutz der Anonymität von Agentenbesitzern gegenüber globalen Namensdiensten.

Dieser Form von Angriffen auf mobile Agenten wurde bisher nicht die notwendige Aufmerksamkeit geschenkt. So bietet beispielsweise das Mobile Agenten System *Ajanta* [6] zwar eine hybride Verschlüsselung von Agentendaten mit den öffentlichen Schlüsseln autorisierter Wirtssysteme, jedoch ist dieser Mechanismus durch den von mir beschriebenen *cut & paste* Angriff verletzbar.

Zu den angedeuteten Schutzmechanismen sind zum Teil bereits Referenzimplementierungen entstanden (Verschlüsselung und Abwehr von *cut & paste* Angriffen) oder sind im Entstehen (sichere Namensdienste auf Basis impliziter Namen).

Literatur

[1] David M. Chess. Security issues in mobile code systems. In Vigna [21], pages 1–14.

[2] Erwin Engeler and Peter Läuchli. *Berechnungstheorie für Informatiker.* Teubner Verlag, Stuttgart, 1988.

[3] William M. Farmer, Joshua D. Guttman, and Vipin Swarup. Security for mobile agents: Authentication and state appraisal. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, volume 1146 of *Lecture Notes in Computer Science*, pages 118–130, September 1996.

[4] FIPS180. Secure Hash Standard. Federal Information Processing Standards Publication 180, U.S. Department of Commerce/National Bureau of Standards, National Technical Information Service, Springfield, Virginia, May 1993.

[5] G. Karjoth, N. Asokan, and C. Gülcü. Protecting the computation results of free–roaming agents. In Rothermel and Hohl [13], pages 195–207.

[6] Neeran M. Karnik and Anand R. Tripathi. Agent server architecture for the Ajanta mobile-agent system. In *Proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '98)*, Las Vegas, July 1998.

[7] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* Discrete Mathematics and its Applications. CRC Press, New York, 1996. ISBN 0-8493-8523-7.

[8] Luc Moreau. Distributed directory service and message routing for mobile agents. Technical Report ECSTR M99/3, Department of Electronics and Computer Science, University of Southampton, November 1999.

[9] Amy L. Murphy and Gian Pietro Picco. Reliable communication for highly mobile agents. In *Proc. First International Symposium on Agent Systems and Applications, and Third International Symposium on Mobile Agents (ASA/MA '99)*, pages 141–150. IEEE Computer Society Press, 1999.

[10] Volker Roth. Scalable and Secure Global Name Services for Mobile Agents. 6th ECOOP Workshop on Mobile Object Systems: Operating System Support, Security and Programming Languages (Cannes, France, June 2000).

[11] Volker Roth and Vania Conan. Encrypting Java Archives and its Application to Mobile Agent Security. In Frank Dignum and Carles Sierra, editors, *Agent Mediated Electronic Commerce: A European Perspective*, volume 1991 of *Lecture Notes in Artifical Intelligence*, pages 232–244. Springer Verlag, Berlin, 2000.

[12] Volker Roth and Mehrdad Jalali. Access Control and Key Management for Mobile Agents. *Computers & Graphics, Special Issue on Data Security in Image Communication and Networks*, 22(4):457–461, 1998.

[13] K. Rothermel and F. Hohl, editors. *Proceedings of the Second International Workshop on Mobile Agents (MA '98)*, volume 1477 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin Heidelberg, September 1998.

[14] RSA Laboratories. Cryptographic message syntax standard. Public Key–Cryptography Standards 7, RSA Laboratories, Redwood City, CA, USA, 1993. Available at URL: ftp://ftp.rsa.com/pub/pkcs/.

[15] RSA Laboratories. Selected attribute types. Public Key–Cryptography Standards 8, RSA Laboratories, Redwood City, CA, USA, 1993. Available at URL: ftp://ftp.rsa.com/pub/pkcs/.

[16] Tomas Sander and Christian F. Tschudin. Protecting mobile agents against malicious hosts. In Vigna [21], pages 44–60.

[17] Peter Sewell, Pawel Wojciechowski, and Benjamin Pierce. Location-independent communication for mobile agents: a two-level architecture. Technical Report 462, Computer Laboratory, University of Cambridge, April 1999.

[18] J. Shoch and J. Hupp. The Worms Programs – Early Experience with Distributed Computing. *Communications of the ACM*, 25(3):172–180, March 1982.

[19] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, Inc., 1992.

[20] Giovanni Vigna. Cryptographic traces for mobile agents. In *Mobile Agents and Security* [21], pages 137–153.

[21] Giovanni Vigna, editor. *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin Heidelberg, 1998.

[22] Jan Vitek and Christian Jensen. *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, volume 1603 of *Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1999.

[23] James E. White. *Mobile Agents*, chapter 18. AAAI/MIT Press, 1997.

[24] Pawel Wojciechowski and Peter Sewell. Nomadic Pict: Language and Infrastructure Design for Mobile Agents. In *Proc. First International Symposium on Agent Systems and Applications, and Third International Symposium on Mobile Agents (ASA/MA '99)*, volume 2, pages 821–826, October 1999.

[25] Bennet S. Yee. A sanctuary for mobile agents. In *Secure Internet Programming* [22], pages 261–273.